Java String intern()

The **Java String class intern()** method returns the interned string. It returns the canonical representation of string.

It can be used to return string from memory if it is created by a new keyword. It creates an exact copy of the heap string object in the String Constant Pool.

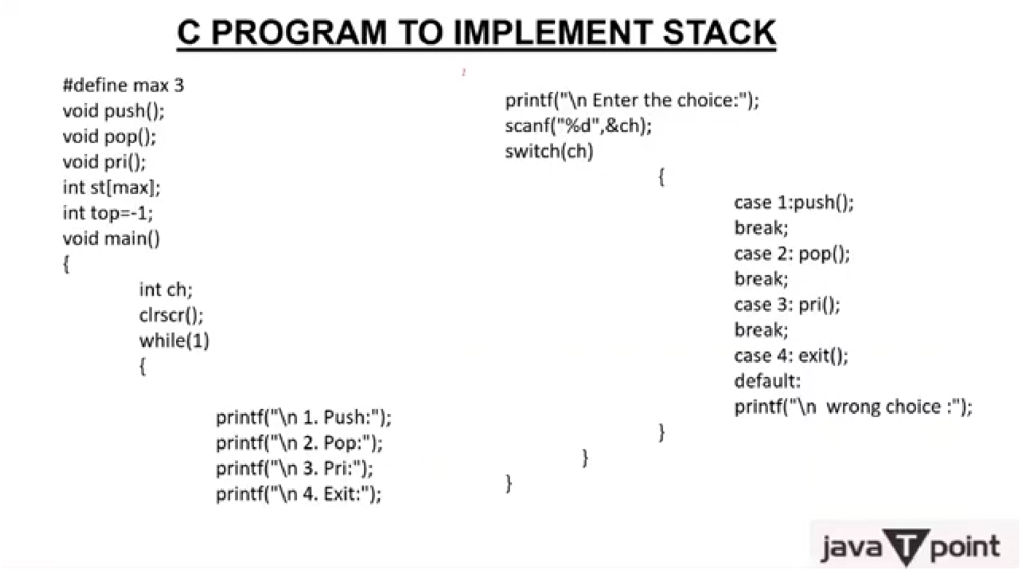
Signature

The signature of the intern() method is given below:

```
public String intern()
```

Returns

interned string



C PROGRAM TO IMPLEMENT STACK

```
#define max 3
void push();
void pop();
void pri();
int st[max];
int top=-1;
void main()
{
        int ch;
        clrscr();
        while(1)
        {

                printf("\n 1. Push:");
                printf("\n 2. Pop:");
                printf("\n 3. Pri:");
                printf("\n 4. Exit:");
```

```
printf("\n Enter the choice:");
scanf("%d",&ch);
switch(ch)
                {
                        case 1:push();
                        break;
                        case 2: pop();
                        break;
                        case 3: pri();
                        break;
                        case 4: exit();
                        default:
                        printf("\n  wrong choice :");
                }
        }
}
```

The need and working of the String.intern() Method

When a string is created in Java, it occupies memory in the heap. Also, we know that the String class is immutable. Therefore, whenever we create a string using the new keyword, new memory is allocated in the heap for corresponding string, which is irrespective of the content of the array. Consider the following code snippet.

```
String str = new String("Welcome to JavaTpoint.");
String str1 = new String("Welcome to JavaTpoint");
```

```
System.out.println(str1 == str); // prints false
```

The println statement prints false because separate memory is allocated for each string literal. Thus, two new string objects are created in the memory i.e. str and str1. that holds different references.

We know that creating an object is a costly operation in Java. Therefore, to save time, Java developers came up with the concept of String Constant Pool (SCP). The SCP is an area inside the heap memory. It contains the unique strings. In order to put the strings in the string pool, one needs to call the **intern()** method. Before creating an object in the string pool, the JVM checks whether the string is already present in the pool or not. If the string is present, its reference is returned.

```
String str = new String("Welcome to JavaTpoint").intern(); // statement - 1
String str1 = new String("Welcome to JavaTpoint").intern(); // statement - 2
System.out.println(str1 == str); // prints true
```

In the above code snippet, the intern() method is invoked on the String objects. Therefore, the memory is allocated in the SCP. For the second statement, no new string object is created as the content of str and str1 are the same. Therefore, the reference of the object created in the first statement is returned for str1. Thus, str and str1 both point to the same memory. Hence, the print statement prints true.

Java String intern() Method Example

FileName: InternExample.java

```
public class InternExample{
public static void main(String args[]){
String s1=new String("hello");
String s2="hello";
String s3=s1.intern();//returns string from pool, now it will be same as s2
System.out.println(s1==s2);//false because reference variables are pointing to different instance
System.out.println(s2==s3);//true because reference variables are pointing to same instance
}}
```

Test it Now

Output:

```
false
true
```

Java String intern() Method Example 2

Let's see one more example to understand the string intern concept.

FileName: InternExample2.java

```
public class InternExample2 {
    public static void main(String[] args) {
        String s1 = "Javatpoint";
        String s2 = s1.intern();
        String s3 = new String("Javatpoint");
        String s4 = s3.intern();
        System.out.println(s1==s2); // True
        System.out.println(s1==s3); // False
        System.out.println(s1==s4); // True
        System.out.println(s2==s3); // False
        System.out.println(s2==s4); // True
        System.out.println(s3==s4); // False
    }
}
```

Test it Now

Output:

```
true
false
true
false
true
false
```

Points to Remember

Following are some important points to remember regarding the intern() method:

1) A string literal always invokes the intern() method, whether one mention the intern() method along with the string literal or not. For example,

```
String s = "d".intern();
String p = "d"; // compiler treats it as String p = "d".intern();
System.out.println(s == p); // prints true
```

2) Whenever we create a String object using the **new** keyword, two objects are created. For example,

```
String str = new ("Hello World");
```

Here, one object is created in the heap memory outside of the SCP because of the usage of the new keyword. As we have got the string literal too ("Hello World"); therefore, one object is created inside the SCP, provided the literal "Hello World" is already not present in the SCP.